JORD

ISO15926 RDS ID REQUIREMENTS & IMPLEMENTATION SPECIFICATION

JORD (Joint Operational Reference Data) Project
enhancing the
PCA Reference Data Service (RDS) Operation
in partnership with FIATECH

Rev	Date	Description	By	Check
V1	8th Aug 2011	Generated during Frederick workshop following earlier Semantic Days working session.	ISG / RdeC	
V2	23rd Aug 2011	Consolidated version published following workshop	ISG	
V3.1	Sep 2011	Formatted to form part of development prototype EndPoint Publishing requirements.	ISG	
V3.2	Jan 2012	Minor update with feedback from EndPoint development, and from iRING RDS/WIP.	ISG	
V4	7th Mar 2012	Enhanced to incorporate wider requirements from Part 6 Ballot & Methodology feedback, including ID / Naming conventions, intended for RDL management tools and production EndPoint.	ISG	
V4.1	22nd Jan 2013	Incorporating latest proposal on UUID's and default ID concept following JORD RDL Manager Tool workshop	ISG	
V5	20th May 2013	Consolidated update covering implementation feedback on namespaces and support for human readable URI's	ISG	
	17th June 2013	Consolidated update and improved explanations.	LH/ISG /HO	
V5.1	1st Nov 2013	Completed consolidated update and updated namespace section	LH	
V5.2	18th Nov 2013	Added detailed namespace and Linked Data Section	LH/HO	
V5.3	20th Dec 2013	Improved to match discussion and decisions from TAB	HO	
V5.4	29th Jan 2013	Explicit UUID requirement	HO	

Executive Summary

ISO-15926, the standard for lifecycle integration and interoperability, is based on highly generic information modeling principles, and has a high dependency on shared reference data. Whilst it supports many *valid* and flexible implementation possibilities, these may not support the full lifecycle capabilities *intended* by the standard. Also, being highly generic and flexible, achieving consensus and comprehensive standard interpretations from first principles is non-trivial in specific business circumstances. The JORD Project is establishing standard practices (processes, usage and mapping methodologies and implementation methods) for management and use of reference data applied in the enhanced PCA Reference Data Services (RDS) Operation.

This document defines fundamental requirements ***and*** implementation conventions adopted by JORD for identification of reference data concerned with both its management and its use. As such this specification forms a part of the requirements for the JORD PCA RDS.

Acknowledgements

JORD Charter Member organizations contributed funding and direction to the project:

Full sponsors
EPIM,
RosEnergoAtom,
Black & Veatch,
CCC,
Hatch
and VNIIAES

Supplementary subscribers
Woodside,
DOW,
Emerson
and Bechtel

TABLE OF CONTENTS

1 JORD RDS - Identity, Identification & Naming - General

A priority of JORD, and for all other projects dependent on PCA RDS, is to establish triple-store /end-point support for content publishing so that all RD-Items are can be resolved as web references, whatever the actual Part8-style-RDF-OWL schema & format of the content.

Ultimately this must support / be supported by ***all*** RDL content lifecycle management needs; editing, import /export, validation, change-management and status meta-data, etc. These lifecycle management needs include those of JORD/PCA Core RDL management services, those of ISO, those of sandboxes and shared RDL's hosted externally or internally to JORD/PCA RDS, and those project / business content information sets and repositories dependent on reference to any RD Items in the federated whole, including the querying and reasoning needs of those different projects and businesses. One feature of this challenge is the need for ID's to be immutable for life, once shared within the federation, with any changes subject to justified need and managed migration of any consequences for businesses affected. These represent as set of "use-cases" for the PCA RDS.

These requirements must be supported non-exclusively, so that the federation of multiple distributed RDL's is recognized, with minimum dependence (if any) on the PCA RDS as a central service, with alternative services provided competitively.

In such a scheme all naming, ID's and designations of RD-Items may ultimately be URI's (or relations involving URI's) linking to the relevant resource. The schemes used for identification must therefore so far as possible satisfy such global W3C / Semantic Web / UN-EDIFACT / CEFACT / CCS standards, IEC11179, IEC11578 and RFC4122 (UUID / GUID) as well as SC4 Procedure Annex SK and specific ISO15926 Part 6 RDS management needs.

This document captures agreed identification requirements for these needs. By design, these requirements are more restrictive than the generic base standards – by agreeing additional conventions for identifiers, the intent is greater manageability and assurance of standardized use.

2 Identification, Naming and Designation in URI's

2.1 ID's and Designators

Firstly, it is necessary to define and distinguish two potentially ambiguous kinds of naming as used within this document, with different requirements:

- **ID = non-human intelligible "name"** - In an ***implementation context*** ID and Name can be used synonymously. Here these are referred to as "ID" and are assumed to be non-human-intelligible, system level and generally hidden from business user interfaces. (Note that although not *intended* for human readability, the more compact and intelligible they are, the more humans will actually be able to recognise, learn and communicate what they are.)

- **Designator = human intelligible "name"** - In the ***business context*** Name and Designation can be used synonymously. Here these are referred to as "Designator" and are assumed to be human interpretable and intended to be exposed in business user and/or modeller and developer interfaces.
    - **User Descriptive Name –** human readable, natural descriptive or abbreviated mnemonic terminology for end-user / domain-expert needs
    - **Developer Coding Name** – human readable, conventional or mnemonic terminology for specific programming /ontology representation needs
    - **Business Encoded Name** – human readable, as either of the above, plus additionally formatted or structurally encoded with explicit semantics beyond simple identity within the designator.
      (Note – eg Tags, Model Numbers, etc *These are extremely common in many business contexts and, whilst supportable, it is recommended that such encoded designations are not relied upon for anything other than identification and validation of identity. Where such encoded designations do exist, it is important also to distinguish between the genuinely unique identification part and the additional non-ID parts, both elements of which may include encoding meaning eg schematic line / cable / node / terminal numbers which are often encoded with many aspects of their specification as well as their actual unique ID.*)

2.2 Name components

Within this document Uniform Resource Identifier (URI), which is used to identify RD-Item in RDF/OWL representations, is considered to be a particular Name type. The Name in the URI styleis composed by concatenation of:

- The <**namespace**> - for example base, domain part of the URI, or explicit context element of a Name, itself globally unique, and
- The <**namefragment**> - for example local, #, unique part of the Name or URI

The <namefragment> part may be either locally unique <**local-name**> eg in the context of the <namespace> part or may be globally unique <**global-name**> in its own right.
(eg a UUID - *see PCA RDS ID Conventions later*).

We will say that Name in the URI style is an **ID** if its <**namefragment**> is an ID, and a **Designator**if its <**namefragment**> is a Designator.

So this gives us four distinct URI styles of ID or Designator *possible.*
1) <namespace>+<global-name-ID>
2) <namespace>+<local-name-ID>
3) <namespace>+<local-name-Designator>
4) *<namespace>+<global-name-Designator>*

Note – In case (1) the namespace is primarily concerned with resolvability and addressability of the resource rather than the uniqueness of its identity. In cases (2) & (3) the namespace is fundamental to the uniqueness context for the identity as well as resolvability and addressability of

the resource. Case (4) is logically possible, but unlikely to be supportable, given the number RD-Items and the number of human naming contexts possible.

Note also that RD-Items include resources which are not only distinct class and individual objects, but also templates, signatures and template patterns, patterns describing graphs composed of other RD-Items, including other templates and patterns. These identification rules need also to be applicable to the naming of graphs.
(See PCA RDS ID Conventions later.)

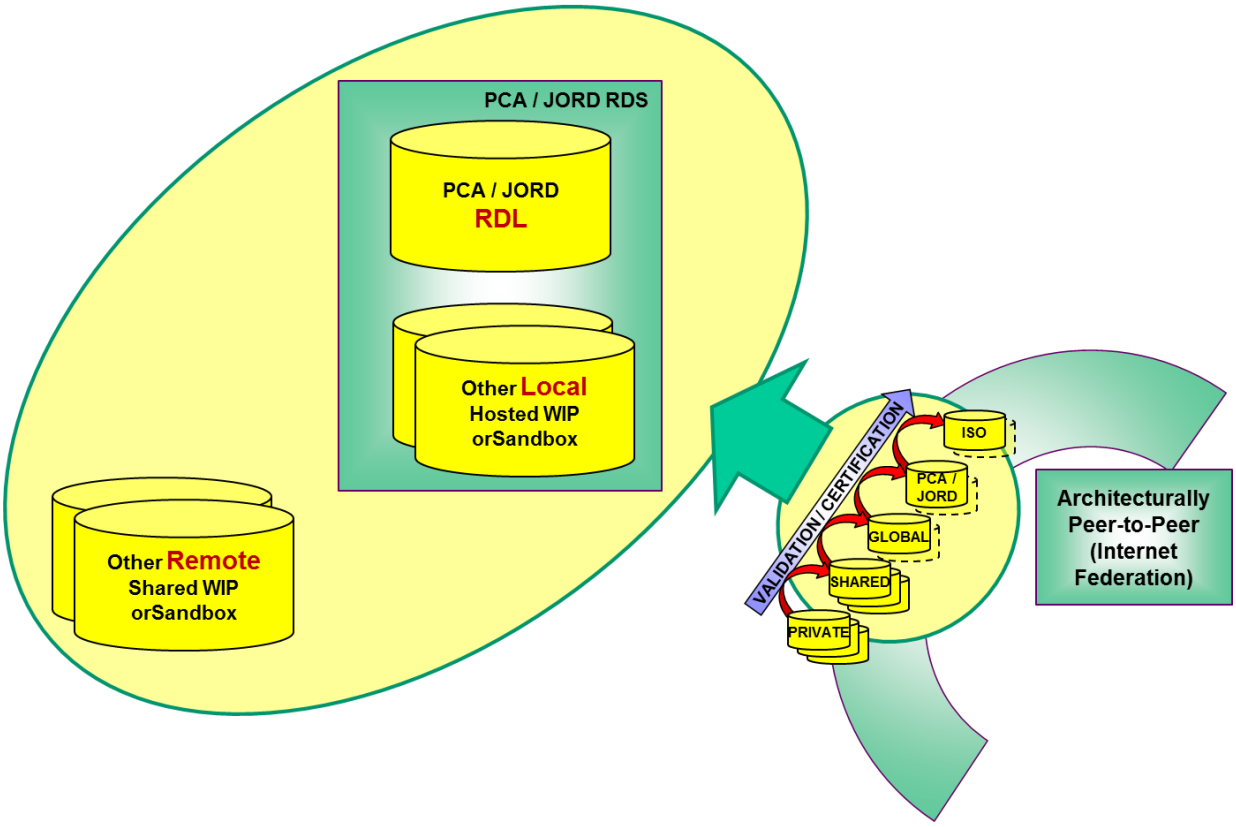
3 General Requirements for IDs and Designators

- Any Name(ID or Designator)<namespace>+<namefragment> shall be globally unique.
- Any Name (ID or Designator)<namespace>+<namefragment> shall identify only one object.
- Any ID ***shall be immutable and persistent*** for all future lifecycles.
- All objects ***shall have at least one***<namespace>+<global-name-ID> type ID.
- Any object may have more than one ID and/or Designator.
  (See PCA RDS management requirements and conventions later.)
- All URIs (ID or Designator type) shall be resolvable to a physical resource.
  (*Note that this will / may involve a resolvability service to access the intended resources – and / or traversing same-as / aliasing / additional ID relations – so service trust is also a key component – see "trust" in the W3C architecture*.)
- Resolvability of URIs (ID or Designator type) may move to different resource(s). (*Note that management requirements will also therefore apply to the moving of resource resolvability*.)
- Version is part of identity for versionable objects. That is for such objects, version identity must be included in naming (ID or Designator) (*See Versionable Objects below*.)

4 ID & Naming Change-Management Use-Cases

Management requirements here imply particular use-cases below, and lifecycle states associated with RD-Items and their use via the RDS.

> *Identification and management requirements for identifiers and versioning are aspects formally covered by ISO15926 Parts 5 & 6. Requirements here which further constrain Part 6 meta-data, and the ISO Procedure replacing Part 5, may be proposed as ballot comments and amendments to those standard parts.*



Case #A New RD-Item created in PCA RDL

A.1 Name (ID and/or Designation) allocated by PCA RDS

Case #B New RD-Item created in local hosted WIP

B.1 Name (ID and/or Designation) allocated by PCA RDS
B.2 Name (ID and/or Designation) allocated by WIP owner

Case #C New RD-Item created in remote WIP

C.1 Name (ID and/or Designation) allocated by WIP owner
C.2 Name (ID and/or Designation) allocated by PCA RDS

Requirement – PCA RDS needs to generate unique ID by design, and validate uniqueness of ID's and Designations by others. (*See PCA RDS ID Conventions later*.)

Case #D Mapping to RD Items

D.1 Any local or remote (or internal to PCA RDL) RD-Item or business data item may map by reference to any other RD-Item

- by Classification (is a / is a member of) or
- by Specialization (is a type of) or
- by Identification (is a local ID/Name or *alias* for) the item with *identical intended semantic*.

One ID/Name may become considered the implementation master / preferred label, others must become slave / "alias" ID's. (See PCA RDS ID Conventions later).

D.2 The ID / Naming of any RD-Item, and the RD-Item itself, shall be immutable and its validity / lifecycle state shall be denoted by meta-data.

Requirement – All mapping and other relations and meta-data attribution shall use triples instantiated using the applicable valid template pattern(s).

Case #E Additional Correlations, Corrections and Consolidation

E.1 Any two RD-Items (different objects with different IDs or Designators) arising in different contexts may subsequently be discovered to have *identical intended semantic* (excluding current lifecycle status meta-data and history).

Requirement - It shall be possible to instantiate a "same as" relation between these two objects.

In the condition where two objects / RD-Item resources are linked by "same as", it may be necessary to denote one object (by meta-data) as the default / master object resource / repository – say the one with the higher level of standardization and certification.

Requirement - It shall be possible to subsequently consolidate "same as" objects into single resources; migrating all relevant relations and updating meta-data accordingly.

E.2 Any RD-Item may be discovered to be erroneous (invalid object type or not meeting the intended semantic) in such a way that it is not corrected simply by additional semantic relations.

E.3 Versionable RD-Items (see versioning) may be superseded by later versions.

Requirement – It shall be possible to retire an erroneous object or a redundant, consolidated or otherwise superseded object, initially by assigning lifecycle meta-data and supercession relations where appropriate, and ultimately by migrating other semantic relations to the later applicable object(s).
Other

Requirement – It shall be possible to search / query / locate / resolve all resources with alternate multiple ID/Names and/or same-as relations irrespective of their physical location. PCA RDS shall include an ID/Naming Registry / Look-up service for all RD-Items visible by mapping relations (D.1) to the PCA RDS. This service shall improve performance, but search & query and resolvability shall not depend on the functioning of this centralized service

Case #F Versionable Objects / RD-Items.

It is a convention that certain objects are considered to be prior or subsequent versions of another. That is they are deemed to retain part of the *same identity*, despite changes to their semantics, but with specific identity defined by their version.

Versionable objects are "named-graphs" – where the content of the graph may change, but the collection retains identity, except for changing the version identity. (For individual objects, supersession is possible, but they are not versioned simply because their involvement in relations changes. It is the collection of objects and relations (a named-graph) that is versioned.

> *Note that the 4D philosophy in 15926-2, intends that strictly ANY & ALL changes in individual relations or properties imply a new lifecycle-part of a whole-lifecycle object, each with distinct identity. One aspect of the JORD standard practices is to permit lifecycle changes in relations (or sets / named-graphs of relations) without creation of new lifecycle-part objects until versions of those sets / graphs represent* ***business-significant*** *lifecycle stages of the whole-lifecycle object. (Refer to the JORD Methodology) This supports the possibility of treating changes to any individual relation (the simplest graph) as business significant, but allows migration to that situation from the more typical situation of the vast majority of implementations to date supported by PCA RDS, RDS/WIP, iRING, iRINGTools, Proteus, etc, where named and versioned graphs represent documents, views, reports, files "about" identifiable subject(s).*

5 PCA RDS Naming Conventions

5.1 Discussion and General Principles

The above represent fundamental requirements and conceptual / logical constraints on possibilities necessary to manage and validate identity, and ensure this can be achieved with genuine lifecycle immutability. The remainder of this specification defines implementation requirements applicable to the PCA RDS Operation *and* to users of those services who expect their own extensions and mappings to the RDL to form part of the iRING federation.

A number of pragmatic principles are adopted:

- Minimising the number of different identification and naming conventions required.
- Minimising the number different naming schemes visibly apparent in human UI's, whilst nevertheless supporting the different human user needs in 2.1 above.
- Minimising changes to existing identifiers of existing content, *and ensuring* any essential retirement or deprecation of existing identifiers and migration to new identifiers or resources are planned and managed in terms of consequences for existing users.
- Minimising the real-time dependence of the iRING federation on the functioning of the PCA RDS.
- Maximising support for widest future semantic web usage where existing 15926 specifications and implementations to date may be more limiting.

5.2 RDS Naming Conventions for Implementation

5.2.1 ID (non-human-intelligible naming) Implementation

UUID's will be adopted as the base for a default ID scheme for any ***new*** identifiers within the whole federation. UUID's can be generated independently by any number of different resource owners according to RFC4122, without the needs for a central ID Generator.

> *Note: UUIDs are documented as part of ISO/IEC 11578:1996 "Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)" and more recently in ITU-T Rec. X.667 | ISO/IEC 9834-8:2005. The IETF has published the Standards-Track, RFC 4122, that is technically equivalent with ITU-T Rec. X.667 | ISO/IEC 9834-8. The proposal is to allow any of the UUID generation options within the standard. (The main choices concern convenience of the generation process and the transparency of reverse engineering the resources that generated the ID – different members of the federation may have different needs here. In the interests of openness we could forbid RFC4122 Option 5, but this is* ***not*** *proposed here. A second assumption here is that the statistical guarantee of uniqueness is sufficiently high, and that in exceptional cases where non-uniqueness arises, the domains will be in sufficiently distinct contexts that these will be immediately apparent and manageable.)*

As a consequence of this, the following mutually exclusive ID conventions will apply immutably to all RD-Items:

a) For those existing RD-Items with an existing RDS/WIP ID (Rnnnnn) this ID will populate a http://posccaesar.org/rdl/defaultRdsId predicate (*R followed by number*)
b) For all existing RD-Items without (a) above, the existing PCA RDS ID (RDSnnnnnn) will populate a http://posccaesar.org/rdl/defaultRdsId predicate (*R followed by D)*
c) For those new items with neither (a) or (b) above, the http://posccaesar.org/rdl/defaultRdsId predicate will be populated with the form R-UUID (*eg. R-f81d4fae-7dec-11d0-a765-00a0c91e6bf6, explicitly following RFC 4112 page 4*)

This convention ensures every RD-Item has a http://posccaesar.org/rdl/defaultRdsId which starts with R and has a form where only its provenance is interpretable, but where uniqueness is global (*independent of this provenance and independent of any explicit context or namespace – see URI's below*). Generally not only is this ID globally unique, but each RD-Item will have only one http://posccaesar.org/rdl/defaultRdsId (*See handling of exceptions and errors below*).

RD-Items may have other ID's used in other contexts, provided applicable Template Signature Patterns are used (implemented as TIP instances, template instances or RDF predicates). Such additional ID's are considered an "*alias*" to the default ID.

5.2.2 Designator (human naming) Implementation

For all RD-Items where business convention demands a human name, the required default name string in English shall populate a http://posccaesar.org/rdl/hasDesignationpredicate.

This is already satisfied for existing PCA RD-Item content, with PCA RDL as the default context for uniqueness and English / ISOxxxx as the default language text encoding.

For all other designations the applicable Template Signature Pattern (TSP) shall be used, to ensure that all other necessary predicates (roles) are also created to define the context for uniqueness, the classification of the type of designator and the symbolic language encoding of the designator string.

Note that human naming needs to satisfy not only human end-user business requirements, but also needs to satisfy human developer / mapper and human library-manager business requirements. (See 2.1 above) And note also that any RD-Item may therefore have multiple designators for different contexts.

5.2.3 URI Implementation

Notwithstanding multiple naming (identification and designation) possibilities satisfying multiple needs above, each physical RD-Item ***is*** a single resolvable web resource with an URI. The expectation is that the URI will be the immutable identifier of the resolvable resource and the

URI's of RD-Items will generally be the references implemented in external data sets, mappings and applications.

ISO15926, PCA, RDS/WIP, iRINGTools and iRING practice to date has been to use the R or RDS numbers as the URI<namefragment> whereas as a semantic-web principle is that for any resource with a business reason forexistenceof an appropriate human intelligible name (designator) should form the URI<namefragment>.

For a single URI of a single resource this represents a conflict. In order to avoid wholesale change of URI schemes already assumed by implementations to date, these two demands need to be seen as additive. Therefore:

- Existing RD-Items will continue with their existing R and RDS number URI's unchanged.
- New RD-Items should use their Designator as the URI<namefragment>, with their URI<namespace>corresponding to the appropriate TSP context-for-uniqueness role.
- Where existing RD-Items require new human-readable Designator based URI's it is necessary to create a new RD-Item which is declared logically the same as the existing RD-Item resource related with an owl:sameAs predicate.
- And, there are however other reasons why an owl:sameAs relations will need to be created. The most common predictable case is where two resources created (erroneously) in different contexts (with similar or different names) within the federation actually represent the same logical RD-Item. Because they are created in different contexts, it will generally only be a process of discovery in business use that reveals the same semantic intent, and the need to add a an owl:sameAs relation.

There are several consequences to the existence of an owl:sameAs relations:

- As well as individual RD-Items being identifiable by its defaultRdsId predicate and any one of multiple alias ID predicates, and any number of Designator predicates, each RD-Item may be addressable viaone or more an owl:sameAs predicate relating it to a logically identical but distinct RD-Item URI.

- Once created, properties of the one logical RD-Item may be associated with any of these multiple distinct RD-Item URI's so it is necessary for queries resolving URI's and returning required properties (predicate values) traverse any an owl:sameAs predicates and return the same required properties from that resource also.

- Work to create these *new* RD-Item and an owl:sameAs content, and the *new* query functionality, is beyond the JORD scope, but the JORD enhanced PCA RDS implementation must support their addition by future project.

- The existence of an owl:sameAs relations – as a result of deliberate intent to support multiple URI conventions, or as a result if discovering semantic errors – implies the need for future consolidation. That is migration of all properties and mappings to the more correct or preferred physical resource, and retirement of the "redundant" resource, in order to simplify ongoing management. However any such consolidation and retirement needs to be part of change management planned with the involvement of business users affected.

Such content consolidation and retirement is beyond the JORD scope, (*except where fixing or retirement of erroneous content is self-evidently part of enhancements to support the JORD methodology and standard practices*).

5.2.4 Uniqueness Context & Namespace Conventions

This section needs to be revised after addition of chapter 6.
There are two distinct needs here:

Firstly, as noted throughout this section (5.2), the context for uniqueness of any Name (ID or Designator) needs to be explicit wherever that context is other than the PCA RDS. Such a context for uniqueness is implemented as the applicable role (predicate) in the appropriate TSP. Where that name is used as the URI<namefragment>, the same context string will form the URI<namespace>, within which the URI<namefragment> is not only unique, but uniquely defined in terms of language encoding.

Secondly, any namespace defined above may be further partitioned for management and usage practicality reasons into sub-domains containing different types of RD-Item content intended for different purposes. The conventions adopted for enhanced PCA RDS Operations are as follows:

General RDL class content not included in one of the further partitions below:
<posccaesar.org></rdl></namefragment> intended for human browsing, and
<posccaesar.org></endpoint></sparql><?= … > intended for SPARQL-querying.

RDL representation of the ISO15926-2 data model:
http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003

Template and Patterns content (including signature and role elements) will use a separate namespace, and will be accessible through a separate endpoint.

5.2.5 Miscellaneous Additional Requirements & Consequences

(a) Naming (ID & Designation) requirements of this specification including human naming (designations) apply equally to Template and Pattern RD-Items including signatures and their role elements.

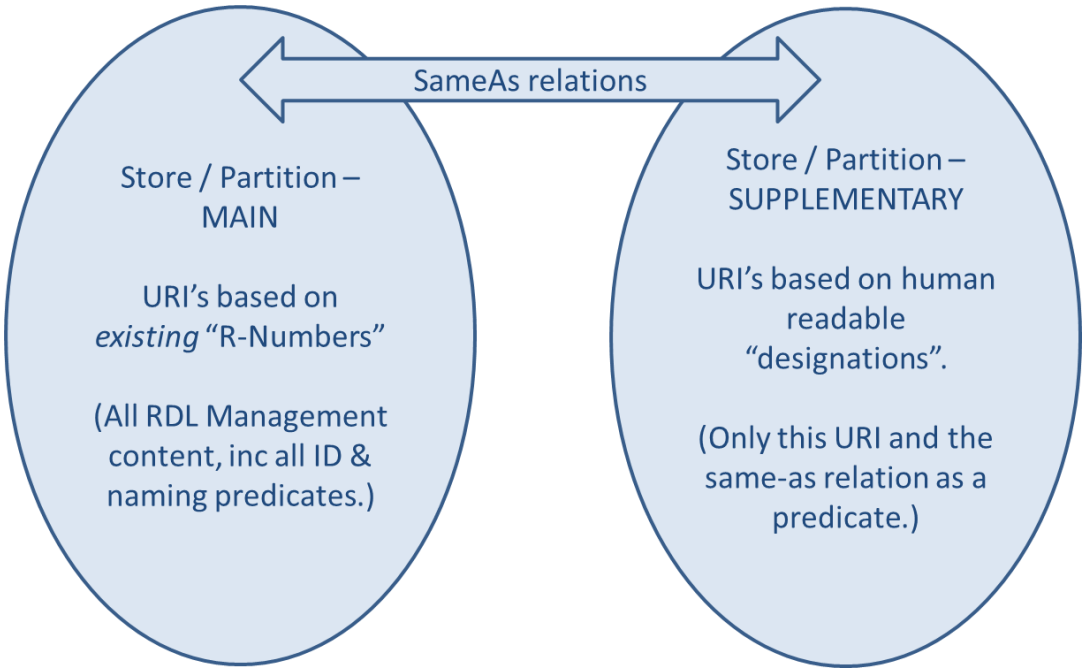
(b) The PCA RDS naming requirements and conventions are applicable to all local RDL's and Sandboxes within the federation. This shall be made clear in the terms for PCA RDS hosting of customer Sandboxes. Ultimately the PCA RDS services shall implement these as rules validating the usage of such sandboxes.

(c) Customer creation of UUID's is not dependent on the PCA RDS Operation, but the PCA RDS Operation may offer a UUID creation service.

(d) Self-sufficient customer users should apply their own SPARQL queries across all available multiple naming (ID & Designator) of all RD-Item URI's and <sameAs> relations. PCA RDS Operation may provide a registry look-up service across multiple naming, alias and <sameAs> relations.

(e) It will be necessary to make a one-shot catch-up update for all RDS/WIP Items with a current Rnnnn ID's as pca:defaultRdsID. This catch-up applies by agreement to all RDS/WIP items published up to the point when the proposed convention is adopted by PCA RDS in JORD Endpoint V3. (*In practice there should be no new RDS/WIP Rnnnn ID's being created since the RDS/WIP server shutdown in November 2012*)

(f) The net result of adopting these conventions on multiple naming and <sameAs> relations will multiply the size of existing triple stores. This may justify additional partitioning between (say) existing default R and RDS number based URI content and new human name(designator)-based URI for semantic-web usage.



6 Namespace, Linked Data and Endpoint

There is a strong need for a convention regarding the use of namespaces in all phases of RDF data. This includes the production data, staging data and data stored in sandboxes.

6.1 Vocabulary

SPARQL – A query language for RDF data.
RDF – A data format used to represent graph data.
OWL – An ontology language written in RDF with strictly defined semantics. Sound and complete.
RDFS – Schema language for RDF with strictly defined semantics. Sound but only complete with extensions.
Resource – An union of all subjects, objects and properties.
Subject – The left most part of a triple
Object – The right most part of a triple
Property – The middle part of a triple
Triple – Data represented as a directional graph. E.g. A B C, where there is an edge between A and C that has the name B.
Fully qualified URI – The URI for a resource including its namespace and local name. E.g.http://example.com/bla/thing
Namespace – The part of a fully qualified URI that is common for a group of resources. E.g.http://example.com/bla/
Local name – The part of a fully qualified URI that is the name for a resource. E.g. thing
Prefix – A short form used to represent a namespace. E.g. "ex" instead of http://example.com. Used together with the local name like this: ex:Thing.
Endpoint – A SPARQL endpoint that can receive, process and respond to a SPARQL query over HTTP.
Production environment – An environment that is used in production. For PCA this includes a production endpoint and production linked data pages. For others this may also include their own production servers and production solutions running in client controlled environments.
Staging environment – The last non-production environment in a chain of environments. Non-production environments simulate production environments and are used to test changes and alterations to a system before these changes are reflected in the production environment.
Subdomain – A domain that is one level below the main domain. Eg. foo is the subdomain in http://foo.example.com
Sandbox – An endpoint where extensions and improvements to the PCA RDL can be stored and published. A sandbox can have two types of content. Type 1 is for hosting data that should make it into the PCA RDL. Type 2 is for hosting data that should not make it into the PCA RDL. It is up to the sandbox user to decide which type of content they have.

6.2 Common Base Namespace Recommendation

All data must use http://data.posccaesar.org as a base namespace if the data is meant to become a part of the PCA RDL.

All data must use http://sandboxName.community.posccaesar.org as a base namespace for data that is not meant to become part of the PCA RDL (sandboxName should be your sandbox name).

For data in a PCA hosted sandbox you are advised to use the above namespace, for data hosted elsewhere you may do as you wish. Should you need to use a namespace that is not based on posccaesar.org in data hosted by PCA you should use an appropriate subdomain according to 6.9.

6.2.1 Justifications

The use of a common base namespace in a separate subdomain means that the DNS record can easily be made to point to a separate server or even a server in another organization should this be required in the future.

A separate common namespace will also mean that there will not be any conflicts with other services running at PCA servers.

Using a separate namespace for content that is not meant to become part of the PCA RDL means that linked data and endpoint discovery will work seamlessly. However should the content be moved into the PCA RDL at a later point the namespace may have to change.

6.3 Production Namespace from day one

All data should use a production namespace regardless of where the data is located if the intention is to bring the data into production in the future.

6.3.1 Justifications

The main reason for having a staging endpoint is to be able to find flaws and incompatibilities before data is brought into a production environment. Tools developed for production data should be able to work on staging data without having to be modified, be that changing the namespace in SPARQL queries or changing application code to reflect new namespaces.

If the same namespace is used in sandboxes, the staging endpoint and the production endpoint then any SPARQL queries run against any of these endpoints will have the same expected result at any of these locations.

If the same namespace is also used in files, then diffing two files can easily be done to determine what changes have been made.

For sandboxes with content that should not move to the PCA RDL, then the namespace should still be a production namespace, and any staging sandboxes below the main sandbox should also use the production namespace. An example of such a sandbox production namespace: http://sand1.community.posccaesar.org/templates/

6.4 Production Namespace list

***PCA RDL content**: http://data.posccaesar.org/rdl/*
***Template definitions**: http://data.posccaesar.org/tpl/*
***P2 data model**:http://data.posccaesar.org/dm/*

***TSP**: http://data.posccaesar.org/ptrn/*

For sandboxes with content that should not move to the RDL, an incomprehensivelist of namespaces is as above where the base namespace (http://data.posccaesar.org/) should be exchanged for the base namespace of the sandbox. Eg. http://sand1.community.posccaesar.org/rdl/

6.4.1 Justifications

For content there is an assumption that similar content would be found in a common namespace. This is true for the OWL, RDF, RDFS namespaces. However, when using human-readable local names, one single namespace would inevitably cause conflicts.

With a logical partitioning of the data into naturally disjoint partitions there is more room to navigate conflicting local names without making membership of a partition unambiguous. Choosing to partition the namespaces according to a physical or philosophical principle, as an example by partitioning templates into Car and Boat templates would mean that the location of an amphibious vehicle would either have to be a member of one partition or be a member of both risking duplicate information and the need for an equivalence relation. Naturally disjoint partitions will exclude such a possibility due to the inherent properties of naturally disjoint partitions.

There will be no need for a second level partition. So there will be no partition within a partition. This is due to both the challenges in keeping partitions naturally disjoint and the class of class system in 15926-2. Except for possible conflicts with part 11, graphs will be considered for grouping naturally disjoint content (within a namespace) so this can be used for instance when using owl:import to just import a part of the RDL.

The namespace for templates could very well be http://data.posccaesar.org/templates/ instead of http://data.posccaesar.org/tpl/. The justification for using “tpl” over “templates” is that “tpl” will be easier to read for a 15926 expert at the cost of novices who will have to be taught the abbreviations. “tpl” will also make it faster to type. This is only significant for formats that do not support prefixes, eg. Excel. With the use of prefixes ordictionary codercompression (eg. zip) there are no benefits with regard to storage or transfer.

6.5 Endpoint Location

The productionendpoint should be located at http://data.posccaesar.org/ . The staging endpoint should be located at http://staging.data.posccaesar.org/ . A sandbox endpoint should be located at http://yourSandboxName.data.posccaesar.org/or at http://yourSandboxName.community.data.posccaesar.org.

To query a particular partition, this limited partition should be available at the partition level in an endpoint. E.g.http://data.posccaesar.org/ or http://data.posccaesar.org/rdl/ where the former is a union of all sub-partitions and the latter is a specific sub-partition. Let P be the ontology in the sub-partitionand D the ontology in the partition above, then D must be a conservative extension of P with regard to the signature of all resources using the namespace of the sub-partition.

6.5.1 Justifications

To make it easier for users to locate the endpoint when looking at a uri and to then navigate through different endpoints to find relevant information there needs to be two hierarchies of endpoints (subdomain and directory based)

The first approach of using subdomains means that an endpoint at a subdomain can have multiple sub-endpoints. E.g.http://staging.mmt.data.posccaesar.org/ could be a staging endpoint for the PCA MMT Special Interest Group, while their regular endpoint could be http://mmt.data.posccaesar.org/.

Data inherently "belongs" to the production endpoint unless stated otherwise. It is implied that a resource, http://data.posccaesar.org/rdl/pump is available in the endpoint at http://data.posccaesar.org/rather than in the staging endpoint or another endpoint such as http://mmt.data.posccaesar.org/. A user needs to explicitly state that another source should be used.

Data that never intends to move to the RDL should "belong" to the production endpoint for that sandbox. Eg. http://sand1.community.posccaesar.org/. The owners of sand1 may require a staging endpoint, which should be located at http://staging.sand1.community.posccaesar.org/.

For queries that do not want or do not need access to the entire library, there is the possibility of querying just a partition of the library. A query run against http://data.posccaesar.org/rdl/?query=…. should return results limited to the rdl partition while a query run against http://data.posccaesar.org/?query=…. should return results from the union all all sub partitions. This is where the notion of conservative extensions come into play in order to guarantee a logically complete query result.

6.5.2 Currently Undefined Behaviour

A query run against a subdomain has undefined behavior with regard to results that can not be computed from this one source alone. An example of this is when a group is improving 5 templates by adding missing information to these templates, a user that asks this group's endpoint (e.g.http://fiveImprovedTemplates.data.posccaesar.org/templates/) for a template that they are not working on would expect to get a result with the template from http://data.posccaesar.org/templates/.

For a case of three endpoints, http://data.posccaesar.org/, http://a.data.posccaesar.org/ and http://b.a.data.posccaesar.org/ there is the following undefined behavior when b.a.data is an implicit improvement on a.data, and a.data is an implicit improvement on data.

1. A resource not explicitly defined in b.a.data is queried for in b.a.data
2. A resource that has been removed in b.a.data is queried for in b.a.data
3. A resource that has been improved* in b.a.data is queried for in b.a.data

For 1, a possible behavior is that the results are a distinct union of b.a.data, a.data and data. For 2, this represents a problem because such a union would return results that have been deleted in the improved version. For 3, improved is meant to mean where a tripple T is replaced by a tripple T´

that retains either the subject or the subject and the property of T. A union of all three data sources would result in both the updated value and the original value being visible at the same time.

*: Where a triple T is replaced by a triple T´ that retains either the subject or the subject and the property of T.

6.6 Context Aware Behaviour at Endpoint Location

GET requests to an endpoint location must return, a GUI for entering queries as default. When "query" is set in the HTTP query string, the returned result must be a result set in accordance with the accept header, or in accordance with special HTTP query parameters defined by Fuseki. A GET request without "query" in the HTTP query string must return the dataset from the endpoint if the accept header is a RDF accept header (e.g. application/rdf+xml) or if there is a Fuseki defined HTTP query parameter to define the content.

6.6.1 Justifications

The endpoint location should be able to deliver a good user experience regardless of who or what interacts with it. And endpoint should be able to handle a human user who wants to write and run a query, a human user who wants to download the entire dataset, a computer that wants to run a query (but not write) and finally a computer that wants to download the entire dataset.

Examples of the above would be:
GET: http://data.posccasar.org/ returns a GUI for a user
GET: http://data.posccaesar.org/?query=abc returns a result set
GET: http://data.posccaesar.org/?query=abc&output=xml returns a result set serialized as XML
GET: http://data.posccaesar.org/?output=xml returns the entire dataset serialized as XML
GET: http://data.posccaesar.org/ with accept header *application/rdf+xml* returns the entire dataset serialized as XML.
GET: http://data.posccaesar.org/?query=abcwith accept header *application/rdf+xml* returns a result set serialized as XML

This is slightly different from the current approach where all SPARQL queries need to be run against http://data.posccaesar.org/sparql with the appended HTTP query string, while any GET request to http://data.posccaesar.org/ always returns the GUI. Now the SPARQL query is sent to the same location as the GUI and the GUI is only returned in the absence of a SPARQL query.

Further work is also required to support the SPARQL 1.1 graph protocol.

6.7 Linked Data

A GET request http://data.posccaesar.org/rdl/pump must return the data for http://data.posccaesar.org/rdl/pump from the http://data.posccaesar.org/ endpoint. A GET request to http://staging.data.posccaesar.org/rdl/pump must return the data for http://data.posccaesar.org/rdl/pump from the http://staging.data.posccaesar.org endpoint. Note that the resource URI remains the same as required by "Production namespace from day one".

6.7.1 Justifications

Linked data is an important step to making it faster and easier for developers and users to start using the data. By default all URIs dereference from the production endpoint. This is a consequence of applying the "Production namespace from day one" requirement. If all data should be dereferenceable within their context (e.g. staging) then they would have to use a special namespace (e.g. a special staging namespace).

For sandboxes with content that should not move to the RDL the base production namespace should be the name of the sandbox, eg. http://sand1.community.posccaesar.org/. This way linked data will work correctly at the cost of having to switch namespaces if the content should one day move into the RDL.

To allow for some context in linked data, the linked data will depend on the user knowing which source they wish to use. This means that a resource http://data.posccaesar.org/rdl/pumpcan be dereferenced to a specific source by providing that source as a subdomain for the request.

To allow for this as seamless behavior for a user, all links on the linked data page must be directed within the same subdomain. Results returned by the SPARQL GUI must also direct to the correct subdomain for the queried endpoint. This means that a query sent to http://staging.data.posccaesar.org/ will return a dataset with, for instance, http://data.posccaesar.org/rdl/core/pump where the href of the anchor tag in the HTML is actually http://staging.data.posccaesar.org/rdl/core/pump while the contents of the tag is http://data.posccaesar.org/rdl/core/pump.

According to the recommendations in the Cool URIs document (link) there should be two separate URLs for each URI. For instance the URI http://data.posccaesar.org/rdl/core/pump would have two separate URLs, http://data.posccaesar.org/data/rdl/core/pump/ for the RDF data and http://data.posccaesar.org/page/rdl/core/pump for the human readable page. These URLs would be available as a HTTP 303 based on the accept header.

All human readable linked data pages must make the user aware of which source is being used and that there may be other sources available (preferably with links to such sources). When a human readable linked data page is requested for a resource that is not available from that source, then the page must specify other possible sources where the resource may be available. It is currently feasible to check all those sources to see if it is available, however that is only feasible as long as there are a limited number of endpoints in the posccaesar.org domain.
For a computer readable linked data page there should not be a list of other sources, however an implementation for such a list would be of interest. The behavior when requesting a resource that is not in the source should be HTTP 404 Not Found, however here a list of other possible sources could also be useful.

6.8 Hash vs. Slash

The recommendation is to use a slash in the namespace.

6.8.1 Justifications

The fragment identifier (eg. http://example.com/rdl#theFragmentIdentifier) is never sent to the server so the server cannot provide linked data for the fragment without resorting to client side

processing (eg. with javascript). When using hash there is an automatic benefit when downloading a file, eg http://example.com/rdl#pumpcan be found in the file at http://example.com/rdl since that is the semantics of fragment identifiers. Using slash this mechanism can be replicated by either having the working directory act as the RDF file when there is no other file specified, eg. http://example.com/rdl/ would return the RDF file while http://example.com/rdl/pump would return the linked data for pump. Still to be decided is what http://example.com/rdl should return.

6.9 Use of External Namespace

Using a company namespace, please add a subdomain to your namespace that you can point to PCA, this way we will be able to provide, 1. Linked data, 2. Full (and partial) download of your data, 3. A sparql endpoint with your data. Ex: Your company is called Example and your domain is example.com, use data.example.com for your base namespace and point your DNS to posccaesar.org (use CNAME).

Example. BigCompany.com uses http://data.BigCompany.com as namespace. Hosting of data is possible at PCA if data.BigCompany.com is DNS CNAME to posccaesar.org.

There are a number of limitations to this approach as changing the DNS records will point all contents within that subdomain to PCA. Also, it still needs to be decided if PCA should have content within its RDL that uses an external namespace.